

Quantifying Differential Privacy under Temporal Correlations

Yang Cao^{*†}, Masatoshi Yoshikawa^{*}, Yonghui Xiao[†], Li Xiong[†]

^{*}Department of Social Informatics, Kyoto University, Kyoto, Japan

Email: {soyo@db.soc., yoshikawa@}i.kyoto-u.ac.jp

[†]Math and Computer Science Department, Emory University, Atlanta, GA

Email: {ycao31, yonghui.xiao, lxiong}@emory.edu

Abstract—Differential Privacy (DP) has received increased attention as a rigorous privacy framework. Existing studies employ traditional DP mechanisms (e.g., the Laplace mechanism) as primitives, which assume that the data are independent, or that adversaries do not have knowledge of the data correlations. However, continuously generated data in the real world tend to be temporally correlated, and such correlations can be acquired by adversaries. In this paper, we investigate the potential privacy loss of a traditional DP mechanism under temporal correlations in the context of continuous data release. First, we model the temporal correlations using Markov model and analyze the privacy leakage of a DP mechanism when adversaries have knowledge of such temporal correlations. Our analysis reveals that the privacy leakage of a DP mechanism may *accumulate and increase over time*. We call it *temporal privacy leakage*. Second, to measure such privacy leakage, we design an efficient algorithm for calculating it in polynomial time. Although the temporal privacy leakage may increase over time, we also show that its supremum may exist in some cases. Third, to bound the privacy loss, we propose mechanisms that convert any existing DP mechanism into one against temporal privacy leakage. Experiments with synthetic data confirm that our approach is efficient and effective.

I. INTRODUCTION

With the development of wearable and mobile devices, vast amount of temporal data generated by individuals are being collected, such as trajectories and web page click streams. The continuous publication of statistics from these temporal data has the potential for significant social benefits such as disease surveillance [4], real-time traffic monitoring [20] and web mining [28]. However, privacy concerns hinder the wider use of these data. To this end, *differential privacy under continual observation* [1] [3] [7] [8] [14] [16] [19] [24] [34] has received increased attention because it provides a rigorous privacy guarantee. Intuitively, differential privacy (DP) [13] ensures that the modification of any single user's data in the database has a "slight" (bounded in ϵ) impact on the change in outputs. The parameter ϵ is defined to be a positive real number to control the privacy level. Larger values of ϵ result in larger privacy leakage.

However, data independence, which is a potential assumption of the DP mechanisms, has been neglected by most previous works in differentially private continuous aggregate release. Recent studies [25] [26] [27] point out that traditional DP may not guarantee the expected privacy on correlated data. The following example shows that the temporal correlations may degrade the expected privacy guarantee of DP.

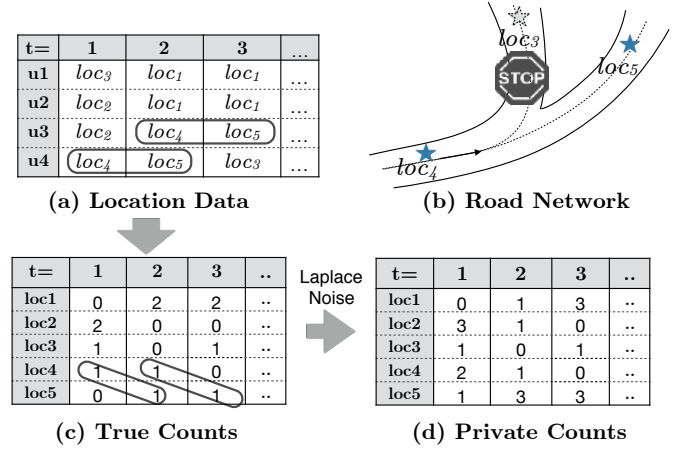


Fig. 1: Differentially Private Continuous Aggregate Release under Temporal Correlations.

Example 1. Consider the scenario of continuously aggregate release illustrated in Figure 1. A trusted server collects users' locations at each time point in Figure 1(a) and continuously publishes aggregate (i.e., the counts of people at each location) in Figure 1(c) with differential privacy. Our goal is to achieve ϵ -DP at each time point t (event-level ϵ -DP [14] [16]) where $t \in [1, T]$. Suppose that each user appears at only one location at each time point. According to the Laplace mechanism [15], adding $\text{Lap}(1/\epsilon)$ noise¹ to perturb each count in Figure 1(c) can achieve ϵ -DP at each time point. However, the expected privacy guarantee may decay due to temporal correlations as follows. Using auxiliary information, such as road network constraints, an attacker may know users' mobility patterns, such as "always arriving at loc₅ after visiting loc₄" (shown in Figure 1(b)), leading to the patterns illustrated in solid lines of Figure 1(c). The temporal correlation due to this road network can be represented as $\Pr(l^t = \text{loc}_5 | l^{t-1} = \text{loc}_4) = 1$ where l^t is the location of a user at time t . That is, given the previous counts of loc₄, an attacker can derive the current count of loc₅. Consequently, because an adversary can perform inference due to such correlations between the two consecutive time points (i.e., as if the same count is released two times), adding $\text{Lap}(1/\epsilon)$ noise to each count guarantees 2ϵ -DP at the time point. Furthermore, let us consider an extreme case of temporal correlation (e.g., a terrible traffic congestion) $\Pr(l^t = \text{loc}_4 | l^{t-1} = \text{loc}_4) = \Pr(l^t = \text{loc}_5 | l^{t-1} = \text{loc}_5) = 1$

¹ $\text{Lap}(b)$ denotes a Laplace distribution with variance $2b^2$.

(i.e., the counts of loc_4 and loc_5 will not change over time). Then, adding $Lap(1/\epsilon)$ noise to each count guarantees $T\epsilon$ -DP at time point T .

It is reasonable to consider that adversaries may obtain the temporal correlations, which commonly exist in our real life and are easily acquired from public information or historical data. In addition to road networks [37], there are countless factors that may cause temporal correlations such as the common patterns characterizing human activities [21] and weather conditions [22].

Few studies in the literature investigated such potential privacy loss of event-level ϵ -DP under temporal correlations as shown in Example 1. A direct method (without finely utilizing the *probability* of correlation) involves amplifying the perturbation in terms of *group differential privacy* [10] [15], i.e., protecting the correlated data as a group. In Example 1, for temporal correlation $\Pr(l^t = loc_5 | l^{t-1} = loc_4) = 1$, we can protect the counts of loc_4 at time $t - 1$ and loc_5 at time t in a group (the sensitivity becomes 2) by increasing the scale of the perturbation to $Lap(2/\epsilon)$ at each time point; for temporal correlation $\Pr(l^t = loc_i | l^{t-1} = loc_i) = 1$, in order to guarantee ϵ -DP at time T , we need to add $Lap(T/\epsilon)$ noise at each time point because the privacy leakage accumulates over time. However, this technique is not suitable for *probabilistic* correlations to finely prevent privacy leakage and may over-perturb the data as a result. For example, regardless of whether $\Pr(l^t = loc_i | l^{t-1} = loc_i)$ is 1 or 0.1, it always protects the correlated data in a bundle.

Although a few studies investigated the issue of differential privacy under *probabilistic* correlations, they are not applicable for *continuous data release* because of the different problem settings. The following two works focused on one-shot data release and different types of correlations. Yang et al. [36] proposed Bayesian differential privacy (BDP), which measures the privacy leakage under probabilistic correlations between tuples, modeled by a Gaussian Markov Random Field without taking time factor into account. Liu et al. [9] proposed *dependent differential privacy* by introducing two parameters of *dependence size* and *probabilistic dependence relationship* between tuples. However, it is not clear whether we can specify them for temporally correlated data. Another line of work [31] [33] [35] has investigated adversaries with knowledge of temporal correlations. They focused on designing new mechanisms for protecting a single user's location privacy extending DP, whereas we attempt to quantify the potential privacy loss of a traditional DP mechanism in the context of continuous aggregate release.

We call the adversary considered in traditional DP with additional knowledge of probabilistic temporal correlations as *adversary \mathcal{T}* . Rigorously quantifying and bounding the privacy leakage against *adversary \mathcal{T}* remains a challenge. Therefore, we attempt to solve the following problems in this paper:

- How do we formalize *adversary \mathcal{T}* and define the privacy loss against *adversary \mathcal{T}* ? (Section III)
- How do we calculate the privacy loss against *adversary \mathcal{T}* ? (Section IV)
- How do we bound the privacy loss against *adversary \mathcal{T}* ? (Section V)

A. Contributions

In this work, for the first time, we quantify and bound the privacy leakage of a DP mechanism due to temporal correlations. Our contributions are summarized as follows.

First, we rigorously define *adversary \mathcal{T}* with temporal correlations that are described by the commonly used Markov model. The temporal correlations include *backward* and *forward* correlations, i.e., $\Pr(l_i^t | l_i^{t-1})$ and $\Pr(l_i^{t-1} | l_i^t)$ where l_i^t denotes the value (e.g., location) of user i at time t . We then define *Temporal Privacy Leakage (TPL)* as the privacy loss of a DP mechanism at time t against *adversary \mathcal{T}* . TPL includes two parts: *Backward Privacy Leakage (BPL)* and *Forward Privacy leakage (FPL)* due to the existence of backward and forward temporal correlations. Our analysis shows that BPL may accumulate from previous privacy leakage and FPL increases with future release. Intuitively, BPL at time t is affected by previously released data and FPL at time t will be affected by future releases. We define *α -differential privacy under temporal correlation*, denoted as α -DP \mathcal{T} , to formalize the privacy guarantee of a DP mechanism against *adversary \mathcal{T}* , i.e., the temporal privacy leakage should be bounded in α . We prove a new form of sequential composition theorem for α -DP \mathcal{T} (different from the traditional sequential composition [30] for ϵ -DP).

Second, we efficiently calculate the temporal privacy leakage under given backward and forward temporal correlations. We translate the calculation of temporal privacy leakage into finding an optimal solution of a *linear-fractional programming* problem. This type of problem can be solved using a simplex algorithm in exponential time. By exploiting the constraints, we propose a polynomial-time algorithm to finely quantify the temporal privacy leakage.

Third, we design private data release algorithms that can be used to convert a traditional DP mechanism into one satisfying α -DP \mathcal{T} . A challenge is that the temporal privacy leakage may increase over time so that α -DP \mathcal{T} is hard to achieve when the length of release time T is unknown. In our first solution, we prove that the supremum of temporal privacy leakage may exist in some cases, and in these cases, we allocate appropriate privacy budgets to make sure the increased temporal privacy leakage will never be greater than α , no matter how long the T is. However, when T is too short so that the accumulation of temporal privacy leakage have not resulted in a significant increase, we may over-perturb the data. The second solution is to exactly achieve α -DP \mathcal{T} at each time point by finely calculating the temporal privacy leakage.

Finally, Experiments with synthetic data confirm the efficiency and effective of our privacy leakage quantification algorithm. We also demonstrate the impact of different degree of temporal correlations on privacy leakage.

II. PRELIMINARIES

A. Differential Privacy.

Differential privacy [13] is a formal definition of data privacy. Let D be a database and D' be a copy of D that is different in any one tuple. D and D' are *neighboring databases*. A differentially private output from D or D' should exhibit little difference.

Definition 1 (ϵ -DP). \mathcal{M} is a randomized mechanism that takes as input D and outputs \mathbf{r} , i.e., $\mathcal{M}(D) = \mathbf{r}$. \mathcal{M} satisfies ϵ -differential privacy (ϵ -DP) if the following inequality is true for any pair of neighboring databases D, D' and all possible outputs \mathbf{r} .

$$\log \frac{\Pr(\mathbf{r}|D)}{\Pr(\mathbf{r}|D')} \leq \epsilon. \quad (1)$$

The parameter ϵ , called the *privacy budget*, represents the degree of privacy offered. Intuitively, a lower value of ϵ implies stronger privacy guarantee and a larger perturbation noise, and a higher value of ϵ implies a weaker privacy guarantee while possibly achieving higher accuracy.

A commonly used method to achieve ϵ -DP is Laplace mechanism, which adds random noise drawn from a calibrated Laplace distribution into the aggregates to be published.

Theorem 1 (Laplace Mechanism). Let $Q : D \rightarrow \mathbb{R}$ be a statistical query on database D . The sensitivity of Q is defined as the maximum L_1 norm between $Q(D)$ and $Q(D')$, i.e., $\Delta = \max_{D, D'} \|Q(D) - Q(D')\|_1$. We can achieve ϵ -DP by adding Laplace noise with scale Δ/ϵ , i.e., $\text{Lap}(\Delta/\epsilon)$.

B. Privacy Leakage.

Let us first discuss the adversaries tolerated by differential privacy, and then formalize privacy leakage w.r.t. such adversaries. Differential privacy is able to protect against the attackers who even have knowledge of all users' data in the database except the one of the targeted victim [17]. Let $i \in \mathcal{U}$ be a user in the database D . Let A_i be an adversary who targets user i and has knowledge of $D_{\mathcal{K}} = D - \{l_i\}$ where $l_i \in [loc_1, \dots, loc_n]$ denotes the data of user i . The adversary A_i observes the private output \mathbf{r} and attempts to guess whether the possible value of user i is loc_j or loc_k where $loc_j, loc_k \in [loc_1, \dots, loc_n]$. We define the privacy leakage of a DP mechanism as follows.

Definition 2 (Privacy Leakage of a DP mechanism against A_i). Let \mathcal{U} be a set of users in the database. Let A_i be an adversary who targets user i and knows all the tuples in the database except the one of user i . The privacy leakage of a differentially private mechanism \mathcal{M} against one A_i and all $A_i, i \in \mathcal{U}$ are defined, respectively, as follows in which l_i and l'_i are two different possible values of user i 's data.

$$PL_0(A_i, \mathcal{M}) \stackrel{\text{def}}{=} \sup_{\mathbf{r}, l_i, l'_i} \log \frac{\Pr(\mathbf{r}|l_i, D_{\mathcal{K}})}{\Pr(\mathbf{r}|l'_i, D_{\mathcal{K}})}$$

$$PL_0(\mathcal{M}) \stackrel{\text{def}}{=} \max_{\forall A_i, i \in \mathcal{U}} PL_0(A_i, \mathcal{M}) = \sup_{\mathbf{r}, D, D'} \log \frac{\Pr(\mathbf{r}|D)}{\Pr(\mathbf{r}|D')}$$

In other words, the privacy budget of a DP mechanism can be considered as a metric of *privacy leakage*. The larger ϵ , the larger the privacy leakage. Hence, we can say that \mathcal{M} satisfies ϵ -DP if $PL_0(\mathcal{M}) \leq \epsilon$. We note that a ϵ' -DP mechanism automatically satisfies ϵ -DP if $\epsilon' < \epsilon$. For convenience, in the following parts of this paper, when we say that \mathcal{M} satisfies ϵ -DP we mean that $PL_0(\mathcal{M}) = \epsilon$.

C. Problem Setting

We attempt to quantify the potential privacy loss a DP mechanism under temporal correlations in the context of *continuous data release* (e.g., releasing private counts at each time as shown in Figure 1). Users in the database, denoted by \mathcal{U} , are generating data continuously. Let $loc = \{loc_1, \dots, loc_n\}$

TABLE I: Summary of Notations.

\mathcal{U}	The set of users in the database
i	The i -th user where $i \in [1, \mathcal{U}]$
loc	Value domain $\{loc_1, \dots, loc_n\}$ of all user's data
$l_i^t, l_i^{t'}$	The data of user i at time t , $l_i^t \in loc$, $l_i^t \neq l_i^{t'}$
D^t	The database at time t , $D^t = \{l_1^t, \dots, l_n^t\}$
\mathcal{M}^t	Differentially private mechanism over D^t
\mathbf{r}^t	Differentially private output at time t
A_i	The adversary who targets user i , considered in traditional DP
A_i^T	Adversary A_i with additional knowledge of temporal correlations
P_i^B	Transition matrix that represents $\Pr(l_i^{t-1} l_i^t)$, i.e., backward temporal correlation, known to A_i^T
P_i^F	Transition matrix that represents $\Pr(l_i^t l_i^{t-1})$, i.e., forward temporal correlation, known to A_i^T
$D_{\mathcal{K}}^t$	The subset of database $D^t - \{l_i^t\}$, known to A_i^T

be all possible values of user's data. We denote the value of user i at time t by l_i^t . A trusted server collects the data of each user into the database $D^t = \{l_1^t, \dots, l_n^t\}$ at each time t (e.g., the columns in Figure 1(a)). A DP mechanism \mathcal{M}^t releases differentially private output \mathbf{r}^t independently at each time t . Our goal is to quantify and bound the potential privacy loss of \mathcal{M}^t against adversaries with knowledge of temporal correlations. We summarize the notations used in this paper in Table I. We note that while we use location data in Example 1, the problem setting is general for temporally correlated data.

Our problem setting is identical to *differential privacy under continual observation* in the literature [1] [3] [7] [8] [14] [16] [19] [24] [34]. In contrast to "one-shot" data release over a static database, the adversaries can observe multiple differentially private outputs, i.e., $\mathbf{r}^1, \dots, \mathbf{r}^t$. There are typically two different privacy goals in the context of continuous data release: *event-level* and *user-level* [14] [16]. The former protects each user's single data point at time t (i.e., the neighboring databases are D^t and $D^{t'}$), whereas the latter protects the presence of a user with all her data on the timeline (i.e., the neighboring databases are $\{D^1, \dots, D^t\}$ and $\{D^{t'}, \dots, D^t\}$). In this work, we mainly study the privacy leakage at a single time point (event-level) under temporal correlations, and we also extend the discussion to user-level privacy by studying the composability of the privacy leakage.

III. ANALYZING PRIVACY LEAKAGE

In the following, we first formalize adversary with temporal correlations in Section III-A. We then define and analyze *temporal privacy leakage* in Section III-B. We provide a new privacy notion of α -DP $_{\mathcal{T}}$ against temporal privacy leakage and prove its composability in Section III-C. Finally, we make a few important observations in Section III-D.

A. Adversary with Knowledge of Temporal Correlations

Markov Chain for Temporal Correlations. The Markov chain (MC) is extensively used in modeling user mobility profiles [21] [29] [31] [37]. For a time-homogeneous first-order MC, a user's current value only depends on the previous one. The parameter of the MC is the *transition matrix*, which describes the probabilities for transition between values. The sum of the probabilities in each row of the transition matrix is 1. A concrete example of transition matrix and time-reversed one for location data is shown in Figure 2. As shown in Figure 2(a), if user i was at loc_3 at the previous time $t - 1$, then the probability of being at loc_1 now (time t)

(a) Transition Matrix $\Pr(l_i^{t-1}|l_i^t)$

		time t-1		
		loc ₁	loc ₂	loc ₃
time t	loc ₁	0.1	0.2	0.7
	loc ₂	0	0	1
	loc ₃	0.3	0.3	0.4

Backward Temporal Correlation P_i^B

(b) Transition Matrix $\Pr(l_i^t|l_i^{t-1})$

		time t		
		loc ₁	loc ₂	loc ₃
time t-1	loc ₁	0.2	0.3	0.5
	loc ₂	0.1	0.1	0.8
	loc ₃	0.6	0.2	0.2

Forward Temporal Correlation P_i^F

Backward Temporal Correlation P_i^B Forward Temporal Correlation P_i^F **Fig. 2: Examples of Temporal Correlations.**

is 0.6; namely, $\Pr(l_i^t = loc_1 | l_i^{t-1} = loc_3) = 0.6$. As shown in Figure 2(b), if user i is at loc_1 now (time t); then, the probability of coming from loc_3 (time $t-1$) is 0.7, namely, $\Pr(l_i^{t-1} = loc_3 | l_i^t = loc_1) = 0.7$. We call the transition matrices in Figure 2(a) and (b) as forward temporal correlation and backward temporal correlation, respectively.

Definition 3 (Temporal Correlations). *The backward and forward temporal correlations between user i 's data l_i^{t-1} and l_i^t are described by transition matrices $P_i^B, P_i^F \in \mathbb{R}^{n \times n}$, representing $\Pr(l_i^{t-1} | l_i^t)$ and $\Pr(l_i^t | l_i^{t-1})$, respectively.*

It is reasonable to consider that the backward and/or forward temporal correlations could be acquired by adversaries. For example, the adversaries can learn them from user's historical trajectories (or the reversed trajectories) by well studied method such as Maximum Likelihood Estimation, or using more sophisticate Markov model [21], or utilizing road network constraints [37]. Also, if the initial distribution of l_i is known (i.e., $\Pr(l_i^1)$), the backward temporal correlation (i.e., $\Pr(l_i^{t-1} | l_i^t)$) can be derived from the forward temporal correlation (i.e., $\Pr(l_i^t | l_i^{t-1})$) by the following Bayesian inference.

$$\Pr(l_i^{t-1} | l_i^t) = \frac{\Pr(l_i^t | l_i^{t-1}) * \Pr(l_i^{t-1})}{\sum_{l_i^{t-1}} \Pr(l_i^t | l_i^{t-1}) * \Pr(l_i^{t-1})}$$

Since estimating temporal correlations from data is beyond the scope of this work, we assume the adversaries' prior knowledge about temporal correlations is given in our framework.

We now define an "updated version" of A_i (in Definition 2) with knowledge of temporal correlations.

Definition 4 (Adversary $_{\mathcal{T}}$). *Adversary $_{\mathcal{T}}$ is a class of adversaries who have knowledge of (1) all other users' data $D_{\mathcal{K}}^t$ at each time t except the one of the targeted victim, i.e., $D_{\mathcal{K}}^t = D^t - \{l_i^t\}$, and (2) backward and/or forward temporal correlations represented as transition matrices P_i^B and P_i^F . We denote Adversary $_{\mathcal{T}}$ who targets user i by $A_i^{\mathcal{T}}(P_i^B, P_i^F)$.*

There are three types of adversary $_{\mathcal{T}}$: (i) $A_i^{\mathcal{T}}(P_i^B, \emptyset)$, (ii) $A_i^{\mathcal{T}}(\emptyset, P_i^F)$, (iii) $A_i^{\mathcal{T}}(P_i^B, P_i^F)$, where \emptyset denotes that the corresponding correlations are not known to the adversaries². For simplicity, we denote types (i) and (ii) as $A_i^{\mathcal{T}}(P_i^B)$ and $A_i^{\mathcal{T}}(P_i^F)$, respectively. We note that $A_i^{\mathcal{T}}(\emptyset, \emptyset)$ is the same as the traditional DP adversary A_i without any knowledge of temporal correlations.

We now show what information $A_i^{\mathcal{T}}$ can derive from the temporal correlations.

Lemma 1. *The adversary $A_i^{\mathcal{T}}$ who has knowledge of P_i^B can derive $\Pr(D^{t-1} | D^t) = \Pr(l_i^{t-1} | l_i^t)$.*

Lemma 2. *The adversary $A_i^{\mathcal{T}}$ who has knowledge of P_i^F can derive $\Pr(D^t | D^{t-1}) = \Pr(l_i^t | l_i^{t-1})$.*

We omit the proofs of the lemmas due to space limitations.

²The adversaries of types (i) and (ii) will not "guess" the missing correlations; otherwise, they fall under type (iii).

B. Temporal Privacy Leakage

We now define the privacy leakage w.r.t. adversary $_{\mathcal{T}}$. For the convenience of analysis, let us assume the length of release time³ is T . The adversary $A_i^{\mathcal{T}}$ observes the differentially private outputs $\mathbf{r}^1, \dots, \mathbf{r}^t, \dots, \mathbf{r}^T$ and attempts to infer the value of user i 's data at time t , namely l_i^t . Similar to Definition 2, we define the privacy leakage in terms of differential privacy but in the context of *differential private under continual observation*, which is described in Section II-C.

Definition 5 (Temporal Privacy Leakage, TPL). *Let $D^{t'}$ be a neighboring database of D^t . Let $D_{\mathcal{K}}^t$ be the tuple knowledge of $A_i^{\mathcal{T}}$. We have $D^{t'} = D_{\mathcal{K}}^t \cup \{l_i^t\}$ and $D^{t'} = D_{\mathcal{K}}^t \cup \{l_i^{t'}\}$ where l_i^t and $l_i^{t'}$ are two different values of user i 's data at time t . Temporal Privacy Leakage (TPL) of \mathcal{M}^t w.r.t. a single $A_i^{\mathcal{T}}$ and all $A_i^{\mathcal{T}}, i \in U$ are defined, respectively, as follows.*

$$TPL(A_i^{\mathcal{T}}, \mathcal{M}^t) \stackrel{\text{def}}{=} \sup_{l_i^t, l_i^{t'}, \mathbf{r}^1, \dots, \mathbf{r}^T} \log \frac{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^T | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^T | l_i^{t'}, D_{\mathcal{K}}^t)}. \quad (2)$$

$$TPL(\mathcal{M}^t) \stackrel{\text{def}}{=} \max_{\forall A_i^{\mathcal{T}}, i \in U} TPL(A_i^{\mathcal{T}}, \mathcal{M}^t) \quad (3)$$

$$= \sup_{D^t, D^{t'}, \mathbf{r}^1, \dots, \mathbf{r}^T} \log \frac{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^T | D^t)}{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^T | D^{t'})}. \quad (4)$$

We first analyze $TPL(A_i^{\mathcal{T}}, \mathcal{M}^t)$ (i.e., Equation (2)) because it is key to solve Equation (3) or (4). We can rewrite $TPL(A_i^{\mathcal{T}}, \mathcal{M}^t)$ as follows because $\mathbf{r}^1, \dots, \mathbf{r}^T$ are published independently by differentially private mechanism $\mathcal{M}^1, \dots, \mathcal{M}^T$.

$$\begin{aligned} \text{Eqn. (2)} &= \sup_{l_i^t, l_i^{t'}, \mathbf{r}^1, \dots, \mathbf{r}^T} \log \frac{\Pr(\mathbf{r}^1 | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\mathbf{r}^1 | l_i^{t'}, D_{\mathcal{K}}^t)} * \dots * \frac{\Pr(\mathbf{r}^T | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\mathbf{r}^T | l_i^{t'}, D_{\mathcal{K}}^t)} \\ &= \underbrace{\sup_{\mathbf{r}^1, \dots, \mathbf{r}^t, l_i^t, l_i^{t'}} \log \frac{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^t | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^t | l_i^{t'}, D_{\mathcal{K}}^t)}}_{\text{backward privacy leakage}} + \underbrace{\sup_{\mathbf{r}^{t+1}, \dots, \mathbf{r}^T, l_i^t, l_i^{t'}} \log \frac{\Pr(\mathbf{r}^{t+1}, \dots, \mathbf{r}^T | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\mathbf{r}^{t+1}, \dots, \mathbf{r}^T | l_i^{t'}, D_{\mathcal{K}}^t)}}_{\text{forward privacy leakage}} \\ &= \underbrace{\sup_{\mathbf{r}^t, l_i^t, l_i^{t'}} \log \frac{\Pr(\mathbf{r}^t | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\mathbf{r}^t | l_i^{t'}, D_{\mathcal{K}}^t)}}_{PL_0(A_i^{\mathcal{T}}, \mathcal{M}^t)} \end{aligned} \quad (5)$$

It is clear that $PL_0(A_i^{\mathcal{T}}, \mathcal{M}^t) = PL_0(A_i, \mathcal{M}^t)$ because PL_0 indicates the privacy leakage w.r.t. one output \mathbf{r} (refer to Definition 2). As annotated in the above equation, we define backward and forward privacy leakage as follows.

Definition 6 (Backward Privacy Leakage, BPL). *The privacy leakage of \mathcal{M}^t caused by $\mathbf{r}^1, \dots, \mathbf{r}^t$ w.r.t. $A_i^{\mathcal{T}}$ is called backward privacy leakage, defined as follows.*

$$BPL(A_i^{\mathcal{T}}, \mathcal{M}^t) \stackrel{\text{def}}{=} \sup_{l_i^t, l_i^{t'}, \mathbf{r}^1, \dots, \mathbf{r}^t} \log \frac{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^t | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^t | l_i^{t'}, D_{\mathcal{K}}^t)}. \quad (6)$$

$$BPL(\mathcal{M}^t) \stackrel{\text{def}}{=} \max_{\forall A_i^{\mathcal{T}}, i \in U} BPL(A_i^{\mathcal{T}}, \mathcal{M}^t). \quad (7)$$

Definition 7 (Forward Privacy Leakage, FPL). *The privacy leakage of \mathcal{M}^t caused by $\mathbf{r}^t, \dots, \mathbf{r}^T$ w.r.t. $A_i^{\mathcal{T}}$ is called forward privacy leakage, defined by follows.*

$$FPL(A_i^{\mathcal{T}}, \mathcal{M}^t) \stackrel{\text{def}}{=} \sup_{l_i^t, l_i^{t'}, \mathbf{r}^t, \dots, \mathbf{r}^T} \log \frac{\Pr(\mathbf{r}^t, \dots, \mathbf{r}^T | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\mathbf{r}^t, \dots, \mathbf{r}^T | l_i^{t'}, D_{\mathcal{K}}^t)}. \quad (8)$$

$$FPL(\mathcal{M}^t) \stackrel{\text{def}}{=} \max_{\forall A_i^{\mathcal{T}}, i \in U} FPL(A_i^{\mathcal{T}}, \mathcal{M}^t). \quad (9)$$

By substituting Equation (6) and (8) into (5), we have

$$TPL(A_i^{\mathcal{T}}, \mathcal{M}^t) = BPL(A_i^{\mathcal{T}}, \mathcal{M}^t) + FPL(A_i^{\mathcal{T}}, \mathcal{M}^t) - PL_0(A_i^{\mathcal{T}}, \mathcal{M}^t). \quad (10)$$

³In this paper, we do not need to know the length of release time in advance.

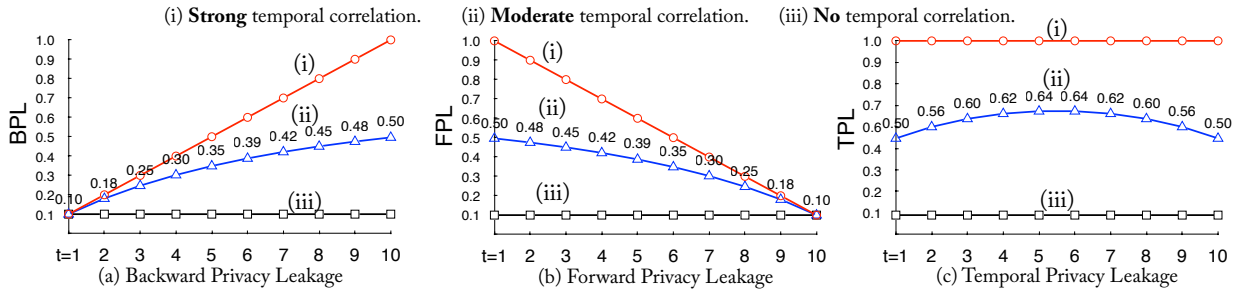


Fig. 3: Example of Temporal Privacy Leakage of $Lap(1/0.1)$ at each time point.

Similarly, by expanding Equation (4) to one resembling Equation (5) and combining it with Equation (7) and (9), we have

$$TPL(\mathcal{M}^t) = BPL(\mathcal{M}^t) + FPL(\mathcal{M}^t) - PL_0(\mathcal{M}^t). \quad (11)$$

Intuitively, BPL and FPL are the privacy leakage w.r.t. the adversaries $A_i^T(P_i^B)$ and $A_i^T(P_i^F)$, respectively. TPL is the privacy leakage w.r.t. $A_i^T(P_i^B, P_i^F)$. In Equation 11, we need to minus $PL_0(\mathcal{M}^t)$ because it is counted in both BPL and FPL. We will show more details in the following analysis.

BPL over time. For BPL, we first expand and simplify Equation (6) by Bayesian theorem and Lemma 1, $BPL(A_i^T, \mathcal{M}^t)$ is equal to

$$\begin{aligned} & \sup_{\mathbf{r}^1, \dots, \mathbf{r}^{t-1}} \log \frac{\sum_{l_i^{t-1}} \Pr(\mathbf{r}^1, \dots, \mathbf{r}^{t-1} | l_i^{t-1}, D_K^{t-1}) \Pr(l_i^{t-1} | l_i^t)}{\sum_{l_i^{t-1'}} \underbrace{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^{t-1} | l_i^{t-1'}, D_K^{t-1})}_{(i) BPL(A_i^T, \mathcal{M}^{t-1})} \underbrace{\Pr(l_i^{t-1'} | l_i^{t'})}_{(ii) P_i^B}} \\ & + \sup_{l_i^t, l_i^{t'}, \mathbf{r}^t} \log \frac{\Pr(\mathbf{r}^t | l_i^t, D_K^t)}{\Pr(\mathbf{r}^t | l_i^{t'}, D_K^t)}. \end{aligned} \quad (12)$$

We now discuss the three annotated terms in the above equation. The first term indicates BPL at the previous time $t-1$, the second term is the backward temporal correlation determined by P_i^B , and the third term is equal to the privacy leakage w.r.t. adversaries in traditional DP (see Definition 2). Hence, BPL at time t depends on (i) BPL at time $t-1$, (ii) the backward temporal correlations, and (iii) the (traditional) privacy leakage of \mathcal{M}^t (which is related to the privacy budget allocated to \mathcal{M}^t). By Equation (12), we know that if $t=1$, $BPL(A_i^T, \mathcal{M}^1) = PL_0(A_i, \mathcal{M}^1)$; if $t > 1$, we have the following, where $\mathcal{L}^B(\cdot)$ is a backward temporal privacy loss function for calculating the accumulated privacy loss.

$$BPL(A_i^T, \mathcal{M}^t) = \mathcal{L}^B(BPL(A_i^T, \mathcal{M}^{t-1})) + PL_0(A_i, \mathcal{M}^t) \quad (13)$$

Equation (13) reveals that the BPL is calculated recursively and may accumulate over time, as shown in Example 2 (Fig.3(a)).

Example 2 (BPL due to previous releases.). Suppose that a DP mechanism \mathcal{M}^t satisfies $PL_0(\mathcal{M}^t) = 0.1$ for each time $t \in [1, T]$, i.e., 0.1-DP at each time point. We now discuss BPL at each time point w.r.t. A_i^T with knowledge of backward temporal correlations P_i^B . In an extreme case, if P_i^B indicates the strongest correlation, say, $P_i^B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, then, at time t , A_i^T knows $l_i^t = l_i^{t-1} = \dots = l_i^1$, i.e., $D^t = D^{t-1} = \dots = D^1$ because of $D^t = \{l_i^t\} \cup D_K^t$ for any $t \in [1, T]$. Hence, the continuous data release $\mathbf{r}^1, \dots, \mathbf{r}^t$ is equivalent to releasing the same database multiple times; the privacy leakage at each time point will accumulate from previous time points and increase linearly (Figure 3(a)(i)). In another extreme case, if there is no backward temporal correlation that is known to A_i^T (e.g.,

for the A_i in Definition 2 or $A_i^T(P_i^F)^4$), the backward privacy leakage at each time point is $PL_0(\mathcal{M}^t)$, as shown in Figure 3(a)(iii). Figure 3(a)(ii) depicts the backward privacy leakage caused by $P_i^B = \begin{pmatrix} 0.8 & 0.2 \\ 0 & 1 \end{pmatrix}$, which can be finely quantified using our method (Algorithm 1) in Section IV.

FPL over time. For FPL, similar to the analysis of BPL, we expand and simplify Equation (6) by Bayesian theorem and Lemma 2, $FPL(A_i^T, \mathcal{M}^t)$ is equal to

$$\begin{aligned} & \sup_{\mathbf{r}^{t+1}, \dots, \mathbf{r}^T} \log \frac{\sum_{l_i^{t+1}} \Pr(\mathbf{r}^{t+1}, \dots, \mathbf{r}^T | l_i^{t+1}, D_K^{t+1}) \Pr(l_i^{t+1} | l_i^t)}{\sum_{l_i^{t+1'}} \underbrace{\Pr(\mathbf{r}^{t+1}, \dots, \mathbf{r}^T | l_i^{t+1'}, D_K^{t+1})}_{(i) FPL(A_i^T, \mathcal{M}^{t+1})} \underbrace{\Pr(l_i^{t+1'} | l_i^{t'})}_{(ii) P_i^F}} \\ & + \sup_{l_i^t, l_i^{t'}, \mathbf{r}^t} \log \frac{\Pr(\mathbf{r}^t | l_i^t, D_K^t)}{\Pr(\mathbf{r}^t | l_i^{t'}, D_K^t)}. \end{aligned} \quad (14)$$

By Equation (14), we know that if $t=T$, $FPL(A_i^T, \mathcal{M}^T) = PL_0(A_i, \mathcal{M}^T)$; if $t < T$, we have the following, where $\mathcal{L}^F(\cdot)$ is a forward temporal privacy loss function for calculating the increased privacy loss due to FPL at the next time.

$$FPL(A_i^T, \mathcal{M}^t) = \mathcal{L}^F(FPL(A_i^T, \mathcal{M}^{t+1})) + PL_0(A_i, \mathcal{M}^t) \quad (15)$$

Equation (15) reveals that FPL is calculated recursively and may increase over time, as shown in Example 3 (Fig.3(b)).

Example 3 (FPL due to future releases.). Considering the same setting in Example 2, we now discuss FPL at each time point w.r.t. A_i^T with knowledge of forward temporal correlations P_i^F . In an extreme case, if P_i^F indicates the strongest correlation, say, $P_i^F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, then, at time t , A_i^T knows $l_i^t = l_i^{t+1} = \dots = l_i^T$, i.e., $D^t = D^{t+1} = \dots = D^T$ because of $D^t = \{l_i^t\} \cup D_K^t$ for any $t \in [1, T]$. Hence, the continuous data release $\mathbf{r}^t, \dots, \mathbf{r}^T$ is equivalent to releasing the same database multiple times; the privacy leakage at time t will increase when every time new release (i.e., $\mathbf{r}^{t+1}, \mathbf{r}^{t+2}, \dots$) happens. For example, we see that contrary to BPL, the FPL at time 1 is the highest (due to future releases at time 1 to 10) while FPL at time 10 is the lowest (since there is no future release with respect to time 10 yet). When \mathbf{r}^{11} is released, all FPL at time $t \in [1, 10]$ will be updated. In another extreme case, if there is no forward temporal correlation that is known to A_i^T (e.g., for the A_i in Definition 2 or $A_i^T(P_i^B)$), then the forward privacy leakage at each time point is $PL_0(\mathcal{M}^t)$, as shown in Figure 3(b)(iii). Figure 3(b)(ii) depicts the forward privacy leakage caused by $P_i^F = \begin{pmatrix} 0.8 & 0.2 \\ 0 & 1 \end{pmatrix}$, which can be finely quantified using our method (Algorithm 1) in Section IV.

⁴Given the current l_i^t and P_i^F , i.e., $\Pr(l_i^t | l_i^{t-1})$, the adversary cannot derive $l_i^t = l_i^{t-1} = \dots = l_i^1$.

Remark 1. The extreme cases shown in Example 2 and 3 are the upper and lower bound of BPL and FPL. Hence, the backward temporal privacy loss function $\mathcal{L}^B(\cdot)$ in Equation (13) and the forward temporal privacy loss function $\mathcal{L}^F(\cdot)$ in Equation (15) satisfy $0 * \cdot \leq \mathcal{L}^B(\cdot) \leq 1 * \cdot$, where \cdot is BPL at the previous time, and $0 * \cdot \leq \mathcal{L}^F(\cdot) \leq 1 * \cdot$, where \cdot is FPL at the next time, respectively.

From Example 2 and 3, we know that: backward temporal correlation (i.e., P_i^B) does not affect FPL, and forward temporal correlation (i.e., P_i^F) does not affect BPL. In other words, adversary $A_i^T(P_i^B)$ only causes BPL; $A_i^T(P_i^F)$ only causes FPL; while $A_i^T(P_i^B, P_i^F)$ poses a risk on both BPL and FPL.

Figure 3(c) shows TPL, which is calculated using BPL and FPL (see Equation (11)). Given P_i^B and P_i^F , finely quantifying TPL is a challenge. We will design a novel algorithm to calculate them efficiently in Section IV.

C. DP under Temporal Correlations and Its Composability

In this section, we define α -DP $_{\mathcal{T}}$ to provide a privacy guarantee against temporal privacy leakage. We prove its sequential composition theorem and discuss the connection between α -DP $_{\mathcal{T}}$ and ϵ -DP in terms of event-level/user-level privacy [14] [16] and w -event privacy [24].

Definition 8 (α -DP $_{\mathcal{T}}$). For all user i in the database, if TPL of \mathcal{M}^t (see Definition 5) is less than or equal to α , we say that \mathcal{M}^t satisfies α -differential privacy under temporal correlation, denoted by α -DP $_{\mathcal{T}}$.

DP $_{\mathcal{T}}$ is an enhanced version of DP on temporal data. If the data are temporally independent (i.e., for all user i , both P_i^B and P_i^F are \emptyset), an ϵ -DP mechanism satisfies ϵ -DP $_{\mathcal{T}}$. If the data are temporally correlated (i.e., existing user i whose P_i^B or P_i^F is not \emptyset), an ϵ -DP mechanism satisfies α -DP $_{\mathcal{T}}$ where $\alpha \geq \epsilon$.

One may wonder, for a sequence of DP $_{\mathcal{T}}$ mechanisms on the timeline, what is the overall privacy guarantee. Suppose that \mathcal{M}^t satisfies ϵ_t -DP and poses risks of BPL and FPL as α_t^B and α_t^F , respectively. That is, \mathcal{M}^t satisfies $(\alpha_t^B + \alpha_t^F - \epsilon_t)$ -DP $_{\mathcal{T}}$ at time t according to Equation (11). We formally define such overall privacy leakage based on Equation (4).

Definition 9 (TPL of a sequence of DP mechanisms). The temporal privacy leakage of DP mechanisms $\{\mathcal{M}^t, \dots, \mathcal{M}^{t+j}\}$ where $j \geq 0$ is defined as follows.

$$TPL(\{\mathcal{M}^t, \dots, \mathcal{M}^{t+j}\}) \stackrel{\text{def}}{=} \sup_{\substack{D^t, \dots, D^{t+j} \\ D^{t'}, \dots, D^{t+j'} \\ \mathbf{r}^1, \dots, \mathbf{r}^T}} \log \frac{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^T | D^t, \dots, D^{t+j})}{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^T | D^{t'}, \dots, D^{t+j'})}$$

It is easy to see that, if $j = 0$, it is event-level privacy; if $t = 1$ and $j = |T - 1|$, it is user-level privacy.

Theorem 2 (Composition under Temporal Correlations). A sequence of DP mechanism $\{\mathcal{M}^t, \dots, \mathcal{M}^{t+j}\}$ satisfies $(\alpha_t^B + \alpha_{t+j}^F + \sum_{k=1}^{j-1} \epsilon_{t+k})$ -DP $_{\mathcal{T}}$ if $j = 1$, and $(\alpha_t^B + \alpha_{t+j}^F + \sum_{k=1}^{j-1} \epsilon_{t+k})$ -DP $_{\mathcal{T}}$ if $j \geq 2$.

We omit the proofs of Theorems 2 due to space limitations.

When $t = 1$ and $j = T - 1$ in Theorem 2, we have the following corollary.

Corollary 1. The temporal privacy leakage of a combined mechanism $\{\mathcal{M}^1, \dots, \mathcal{M}^T\}$ is $\sum_{k=1}^T \epsilon_k$.

It shows that temporal correlations do not affect the user-level privacy (i.e., protecting all the data on the timeline of each user), which is in line with the idea of group differential privacy: protecting all the correlated data in a bundle.

We now compare the privacy guarantee between DP and DP $_{\mathcal{T}}$. As we mentioned in Section II-C, there are typically two privacy notions in continuous data release: *event-level* and *user-level* [14] [16]. Recently, *w-event* privacy [24] is proposed to merge the gap between event-level and user-level privacy. It protects the data in any w -length sliding window by utilizing the following sequential composition theorem of DP.

Theorem 3 (Sequential composition on independent data [30]). Suppose that \mathcal{M}^t satisfies ϵ_t -DP for each $t \in [1, T]$. A combined mechanism $\{\mathcal{M}^1, \dots, \mathcal{M}^{t+j}\}$ satisfies $\sum_{k=1}^{k=j} \epsilon_{t+k}$ -DP.

Suppose that \mathcal{M}^t satisfies ϵ -DP for each $t \in [1, T]$. According to Theorem 3, it achieves $T\epsilon$ -DP on user-level and $w\epsilon$ -DP on w -event level. We compare the privacy guarantee on independent data and temporally correlated data as follows.

TABLE II: The privacy guarantee of ϵ -DP mechanisms.

Privacy Notion \ Data	independent	temporally correlated
event-level [14] [16]	ϵ -DP	α -DP $_{\mathcal{T}}$ ($\alpha \geq \epsilon$)
w -event [24]	$w\epsilon$ -DP	see Theorem 2
user-level [14] [16]	$T\epsilon$ -DP	$T\epsilon$ -DP $_{\mathcal{T}}$ (by Corollary 1)

It reveals that temporal correlations may blur the boundary between event-level privacy and user-level privacy. Considering the examples shown in Figure 3, if no temporal correlation exists, \mathcal{M}^{10} satisfies 0.1-DP on event-level and a combined mechanism $\{\mathcal{M}^1, \dots, \mathcal{M}^{10}\}$ satisfies 1-DP on user-level. However, due to the strongest temporal correlations, \mathcal{M}^{10} satisfies 1-DP $_{\mathcal{T}}$ on event-level and a combined mechanism $\{\mathcal{M}^1, \dots, \mathcal{M}^{10}\}$ also satisfies 1-DP $_{\mathcal{T}}$ on user-level. Essentially, it is because the adversaries may infer $\{D^1, \dots, D^T\}$ (user-level) from D^t (event-level) using temporal correlations.

D. Discussion

We make a few important observations regarding our privacy analysis.

First, the temporal privacy leakage is defined in a personalized way. That is, the privacy leakage may be different for users with distinct temporal patterns (i.e., P_i^B and P_i^F). We define the overall temporal privacy leakage as the maximum one for all users, so that α -DP $_{\mathcal{T}}$ is compatible with the traditional ϵ -DP mechanism (using one parameter to represent the overall privacy level) and we can convert a traditional DP mechanism to bound the temporal privacy leakage. On the other hand, our definitions also can be compatible with personalized differential privacy (PDP) mechanisms [23], in which the personalized privacy budgets, i.e., a vector $[\epsilon_1, \dots, \epsilon_n]$, are allocated to each user. In other words, we can convert a PDP mechanism to bound the temporal privacy leakage for each user.

Second, in this paper, we focus on the temporally correlated data and assume that the adversary has knowledge of temporal correlations modeled by Markov chain. However, it is possible that the adversary has knowledge about more sophisticated temporal correlation model or other types of correlations,

such as user-user correlations modeled by Gaussian Markov Random Field in [36]. Our contributions in this work can serve as primitives for quantifying the privacy risk under more advanced adversarial knowledge.

IV. CALCULATING TEMPORAL PRIVACY LEAKAGE

In this section, we design algorithms for computing backward privacy leakage (BPL) and forward privacy leakage (FPL). We first show that both of them can be translated into the optimal solution of a *linear-fractional programming* problem [2]. Traditionally, this type of problem can be solved by simplex algorithm [11] in exponential time. By exploiting the constraints in this problem, we then design a method to solve it in polynomial time.

A. Problem formulation

According to the privacy analysis of BPL and FPL in Section III-B, we need to solve the backward and forward temporal privacy loss functions $\mathcal{L}^B(\cdot)$ and $\mathcal{L}^F(\cdot)$ in Equations (13) and (15), respectively. By observing the structure of the first term in Equations (12) and (14), we can see that the calculations for recursive functions $\mathcal{L}^B(\cdot)$ and $\mathcal{L}^F(\cdot)$ are virtually in the same way. They calculate the increment of the input values (the previous BPL or the next FPL) based on temporal correlations (backward or forward). Although different degree of correlations result in different privacy loss functions, the methods for analyzing them are the same.

We now quantitatively analyze the temporal privacy leakage. In the following, we demonstrate the calculation of $\mathcal{L}^B(\cdot)$. The first term of Equation (12), i.e., $\mathcal{L}^B(\text{BPL}(A_i^T, \mathcal{M}^{t-1}))$ is as follows.

$$\sup_{\substack{l_i^{t-1}, l_i^{t'} \\ \mathbf{r}^1, \dots, \mathbf{r}^{t-1}}} \log \frac{\sum_{l_i^{t-1}} \Pr(\mathbf{r}^1, \dots, \mathbf{r}^{t-1} | l_i^{t-1}, D_K^{t-1}) \Pr(l_i^{t-1} | l_i^{t'})}{\sum_{l_i^{t-1'}} \underbrace{\Pr(\mathbf{r}^1, \dots, \mathbf{r}^{t-1} | l_i^{t-1'}, D_K^{t-1})}_{\text{BPL}(A_i^T, \mathcal{M}^{t-1})} \underbrace{\Pr(l_i^{t-1'} | l_i^{t'})}_{P_i^B}} \quad (16)$$

We now simplify the notations in the above formula. Let two arbitrary (different) rows in P_i^B be vectors $\mathbf{q} = (q_1, \dots, q_n)$ and $\mathbf{d} = (d_1, \dots, d_n)$. For example, suppose that \mathbf{q} is the first row in the transition matrix of Figure 2(b). Then, the elements in \mathbf{q} are: $q_1 = \Pr(l_i^{t-1} = \text{loc}_1 | l_i^t = \text{loc}_1)$, $q_2 = \Pr(l_i^{t-1} = \text{loc}_2 | l_i^t = \text{loc}_1)$, $q_3 = \Pr(l_i^{t-1} = \text{loc}_3 | l_i^t = \text{loc}_1)$, etc. Let $\mathbf{x} = (x_1, \dots, x_n)^T$ be a vector whose elements indicate $\Pr(\mathbf{r}^1, \dots, \mathbf{r}^{t-1} | l_i^{t-1}, D_K^{t-1})$ with distinct values of $l_i^{t-1} \in \text{loc}$, e.g., x_1 denotes $\Pr(\mathbf{r}^1, \dots, \mathbf{r}^{t-1} | l_i^{t-1} = \text{loc}_1, D_K^{t-1})$. We obtain the following by expanding $l_i^{t-1}, l_i^{t-1'} \in \text{loc}$ in (16).

$$\begin{aligned} \mathcal{L}^B(\text{BPL}(A_i^T, \mathcal{M}^{t-1})) &= \sup_{\mathbf{q}, \mathbf{d} \in P_i^B} \log \frac{q_1 x_1 + \dots + q_n x_n}{d_1 x_1 + \dots + d_n x_n} \\ &= \sup_{\mathbf{q}, \mathbf{d} \in P_i^B} \log \frac{\mathbf{q}\mathbf{x}}{\mathbf{d}\mathbf{x}} \end{aligned}$$

Next, we formalize the problem and constraints. Suppose that $\text{BPL}(A_i^T, \mathcal{M}^{t-1}) = \alpha_{t-1}^B$. According to the definition of BPL (as the supremum), for any $x_j, x_k \in \mathbf{x}$, we have $e^{-\alpha_{t-1}^B} \leq \frac{x_j}{x_k} \leq e^{\alpha_{t-1}^B}$. Considering \mathbf{x} as the variable vector and \mathbf{q}, \mathbf{d} as the coefficient vectors, $\mathcal{L}^B(\alpha_{t-1}^B)$ is equal to the logarithm of the objective function (17) in the following maximization problem (17)~(19).

$$\text{maximize} \quad \frac{\mathbf{q}\mathbf{x}}{\mathbf{d}\mathbf{x}} \quad (17)$$

$$\text{subject to} \quad e^{-\alpha_{t-1}^B} \leq \frac{x_j}{x_k} \leq e^{\alpha_{t-1}^B}, \quad (18)$$

$$0 < x_j < 1 \text{ and } 0 < x_k < 1, \quad (19)$$

$$\text{where } x_j, x_k \in \mathbf{x}, j, k \in [1, n].$$

The above is a form of *linear-fractional programming* [2], where the objective function is a ratio of two linear functions and the constraints are linear inequalities or equations. A linear-fractional programming problem can be converted into a sequence of linear programming problems [2] and then solved using the simplex algorithm [11] in time $O(2^n)$. When n is large, the computation is time consuming.

Bounding the objective function by constraints. We now investigate a more efficient method to solve this problem. From Inequalities (18) and (19), we know that the feasible region of the constraints are not empty and bounded; hence, an optimal solution exists. By exploiting the constraints, we prove the following theorem, which enables the optimal solution to be found in time $O(n^2)$.

We define some notations that will be frequently used in the following parts of this paper. Suppose that the variable vector \mathbf{x} consists of two parts (subsets): \mathbf{x}^+ and \mathbf{x}^- . Let the corresponding coefficients vectors be $\mathbf{q}^+, \mathbf{d}^+$ and $\mathbf{q}^-, \mathbf{d}^-$. Let $\mathbf{q} = \sum \mathbf{q}^+$ and $\mathbf{d} = \sum \mathbf{d}^+$. For example, suppose that $\mathbf{x}^+ = [x_1, x_3]$ and $\mathbf{x}^- = [x_2, x_4, x_5]$. Then, we have $\mathbf{q}^+ = [q_1, q_3]$, $\mathbf{d}^+ = [d_1, d_3]$, $\mathbf{q}^- = [q_2, q_4, q_5]$, and $\mathbf{d}^- = [d_2, d_4, d_5]$. In this case, $\mathbf{q} = \mathbf{q}^+ + \mathbf{q}^-$ and $\mathbf{d} = \mathbf{d}^+ + \mathbf{d}^-$.

Theorem 4. *If the following Inequalities (20) and (21) are satisfied, the maximum value of the objective function in the problem (17)~(19) is $\frac{q(e^{\alpha_{t-1}^B} - 1) + 1}{d(e^{\alpha_{t-1}^B} - 1) + 1}$.*

$$\frac{q_j}{d_j} > \frac{q(e^{\alpha_{t-1}^B} - 1) + 1}{d(e^{\alpha_{t-1}^B} - 1) + 1}, \quad \forall j \in [1, n] \text{ where } q_j \in \mathbf{q}^+, d_j \in \mathbf{d}^+ \quad (20)$$

$$\frac{q_k}{d_k} \leq \frac{q(e^{\alpha_{t-1}^B} - 1) + 1}{d(e^{\alpha_{t-1}^B} - 1) + 1}, \quad \forall k \in [1, n] \text{ where } q_k \in \mathbf{q}^-, d_k \in \mathbf{d}^- \quad (21)$$

Proof: See Appendix A. ■

As we mentioned previously, the calculations of $\mathcal{L}^B(\cdot)$ and $\mathcal{L}^F(\cdot)$ are identical. Therefore, given a transition matrix P_i^B (or P_i^F) and the previous BPL (or the next FPL), the increment of the backward (or forward) privacy loss is the maximum value in the above theorem for any two rows \mathbf{q} and \mathbf{d} in P_i^B (or P_i^F). We denote $\text{FPL}(A_i^T, \mathcal{M}^{t+1})$ by α_{t+1}^F .

$$\mathcal{L}^B(\alpha_{t-1}^B) = \max_{\mathbf{q}, \mathbf{d} \in P_i^B} \log \frac{q(e^{\alpha_{t-1}^B} - 1) + 1}{d(e^{\alpha_{t-1}^B} - 1) + 1} \quad (22)$$

$$\mathcal{L}^F(\alpha_{t+1}^F) = \max_{\mathbf{q}, \mathbf{d} \in P_i^F} \log \frac{q(e^{\alpha_{t+1}^F} - 1) + 1}{d(e^{\alpha_{t+1}^F} - 1) + 1} \quad (23)$$

Corollary 2. *If Inequalities (20) and (21) are satisfied, we have $q > d$ and $q_j > d_j$ in which $q_j \in \mathbf{q}^+$ and $d_j \in \mathbf{d}^+$.*

We omit the proof due to space limitations.

Now, we simply examine the $\mathcal{L}^B(\cdot)$ and $\mathcal{L}^F(\cdot)$ in Equations (22) and (23). First, we have $0 \leq \mathcal{L}^B(\alpha_{t-1}^B)$ and $0 \leq \mathcal{L}^F(\alpha_{t+1}^F)$ because of $q > d$. Second, when \mathbf{q} and \mathbf{d} have the largest difference (e.g., $\mathbf{q} = (1, 0), \mathbf{d} = (0, 1)$ and hence $q = 1, d = 0$), it follows that $\mathcal{L}^B(\alpha_{t-1}^B) \leq \alpha_{t-1}^B$ and $\mathcal{L}^F(\alpha_{t+1}^F) \leq \alpha_{t+1}^F$. Therefore, it is in accordance with Remark 1. The advantage of Equations (22) and (23) is being able to finely quantify BPL and FPL w.r.t. arbitrary P_i^B and P_i^F .

B. Privacy Leakage Quantification Algorithm

The next question is how do we find q and d (or q^+ and d^+) that gives the maximum objective function. Inequalities (20) and (21) in Theorem 4 are sufficient conditions for obtaining such optimal value. Corollary 2 gives necessary conditions for satisfying Inequalities (20) and (21). Based on the above analysis, we design Algorithm 1 for computing BPL or FPL.

Algorithm 1: Finding BPL or FPL

Input: P_i (P_i^B or P_i^F); α (α_{t-1}^B or α_{t+1}^F); ϵ_t (i.e., $PL_0(\mathcal{M}^t)$).
Output: BPL at time t or FPL at time t

```

1  $\mathcal{L}_i \leftarrow 0$  //the value of Equation (22) or (23)
2 foreach two rows  $q, d \in P_i$  do
3   foreach  $q_j \in q, d_j \in d$  do
4     if  $q_j > d_j$  then add  $q_j$  to  $q^+$ ; add  $d_j$  to  $d^+$ 
5   update  $\leftarrow$  false;
6   do //find  $q, d$  by Theorem 4 and Corollary 2
7      $q \leftarrow \sum q^+; d \leftarrow \sum d^+$  //satisfy Inequality (21)
8     foreach  $q_j \in q^+, d_j \in d^+$  do
9       //for satisfying Inequality (20)
10      if  $q_j/d_j \leq (q * (e^\alpha - 1) + 1) / (d * (e^\alpha - 1) + 1)$ 
11      then  $q^+ \leftarrow q^+ - q_j; d^+ \leftarrow d^+ - d_j; \text{update} \leftarrow \text{true};$ 
12   while update
13   if  $\mathcal{L}_i < \log \frac{q * (e^\alpha - 1) + 1}{d * (e^\alpha - 1) + 1}$  then  $\mathcal{L}_i \leftarrow \log \frac{q * (e^\alpha - 1) + 1}{d * (e^\alpha - 1) + 1}$ 
14 return  $\mathcal{L}_i + \epsilon_t$  //by Equation (13) or (15)
```

Computing BPL or FPL by solving the linear-fractional programming. According to the definition of BPL and FPL, we need to return the maximum privacy leakage (Line 12) w.r.t. any two rows in the given transition matrix (Lines 2). Lines 3~11 are to solve one linear-fractional programming problem (17)~(19) w.r.t two specific rows in the transition matrix. In Lines 3 and 4, we divide the variable vector x into two parts according to Corollary 2, which gives the necessary condition for finding the maximum solution: if the coefficients $q_j \leq d_j$, they are not in q^+ and d^+ that satisfy Inequalities (20) and (21). In other words, if $q_j > d_j$, they are “candidates” in q^+ and d^+ that gives the maximum objective function. In Lines 5 to 11, we further check q^+ and d^+ whether they satisfy Inequalities (20) and (21). According to Line 7, it is clear that any subset of q^+ and d^+ automatically satisfy Inequality (21). In Lines 8~10, we remove the pairs $q_j \in q^+$ and $d_j \in d^+$ that do not satisfy Inequality (20). Note that the values of q and d (recall that $q = \sum q^+$ and $d = \sum d^+$) will be recalculated due to such “update” (deletion in Line 10). If q and d are updated, we need to recheck each pair of q_j and d_j in the current set of q^+ and d^+ until every pair of them satisfies Inequality (20).

A subtle question may arise regarding such “update”. In Lines 8~10, if several pairs of q_j and d_j do not satisfy Inequality (20), say, $\{q_1, d_1\}$ and $\{q_2, d_2\}$, one may wonder if it is possible that, after removing $\{q_1, d_1\}$ from q^+ and d^+ , Inequality (20) can be satisfied for $\{q_2, d_2\}$ due to the update of q and d , i.e., $\frac{q_2}{d_2} > \frac{(q-q_1) * (e^\alpha - 1) + 1}{(d-d_1) * (e^\alpha - 1) + 1}$. We show that this is not possible. If $\frac{q_1}{d_1} \leq \frac{q * (e^\alpha - 1) + 1}{d * (e^\alpha - 1) + 1}$, we have $\frac{q * (e^\alpha - 1) + 1}{d * (e^\alpha - 1) + 1} \leq \frac{(q-q_1) * (e^\alpha - 1) + 1}{(d-d_1) * (e^\alpha - 1) + 1}$. Hence, $\frac{q_2}{d_2} \leq \frac{q * (e^\alpha - 1) + 1}{d * (e^\alpha - 1) + 1} \leq \frac{(q-q_1) * (e^\alpha - 1) + 1}{(d-d_1) * (e^\alpha - 1) + 1}$. Therefore, we can remove multiple pairs of q_j and d_j that do not satisfy Inequality (20) at one time (Lines 8~10).

It is easy to know that the update will be terminated with non-empty q^+ and d^+ . If only one element is left in each of q^+ and d^+ , say, q_n and d_n , respectively, then Inequality (20) is true (i.e., $\frac{q_n}{d_n} > \frac{q_n * (e^\alpha - 1) + 1}{d_n * (e^\alpha - 1) + 1}$) because of $q_n > d_n$.

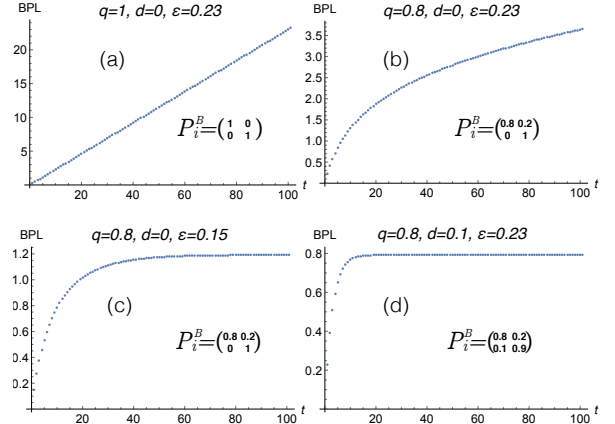


Fig. 4: Examples of the maximum BPL over time.

Algorithm complexity. The time complexity for solving one linear-fractional programming problem (Lines 3~11) w.r.t. two specific rows of the transition matrix is $O(n^2)$ because Line 9 may iterate $n * (n - 1)$ times in the worst case. The overall time complexity of Algorithm 1 is $O(n^4)$.

V. BOUNDING TEMPORAL PRIVACY LEAKAGE

In this section, we design private data release algorithms that can be used to convert a traditional DP mechanism into one satisfying α -DP $_{\mathcal{T}}$ by allocating calibrated privacy budgets.

We first investigate the upper bound of BPL and FPL. We have demonstrated that BPL and FPL may accumulate and increase over time in Figure 3. A natural question is that: is there a limit of BPL and FPL over time. For \mathcal{M}^t that satisfies ϵ -DP at each $t \in [1, T]$, we know that a loose upper bound of BPL or FPL over time T is $T\epsilon$ according to Remark 1. When T is unknown, giving the upper bound of BPL or FPL is a challenge.

Theorem 5. Given a transition matrix P_i^B (or P_i^F) representing temporal correlation, let q and d be the ones that give the maximum value in Equation (22) (or Equation (23)). For \mathcal{M}^t that satisfies ϵ -DP at each $t \in [1, T]$, there are four cases regarding the supremum of BPL (or FPL) over time.

$$\begin{cases} \log \frac{\sqrt{4de^\epsilon(1-q) + (d+qe^\epsilon-1)^2} + d+qe^\epsilon-1}{2d} & d \neq 0 \\ \log \frac{(1-q)e^\epsilon}{1-qe^\epsilon} & d = 0 \text{ and } q \neq 1 \text{ and } \epsilon \leq \log(1/q) \\ \text{not exist} & d = 0 \text{ and } q \neq 1 \text{ and } \epsilon > \log(1/q) \\ \text{not exist} & d = 0 \text{ and } q = 1 \end{cases}$$

The above theorem is applicable for both BPL and FPL because the calculation of BPL and FPL is the same. According to the previous analysis, we can consider that the growth of BPL and FPL is in the same manner but in the reversed directions on the timeline (see Figure 3(a)(b)).

We omit the proof due to space limitations.

Example 4 (The supremum of the increased BPL over time.). Suppose that \mathcal{M}^t satisfies ϵ -DP at each time point. In Figure 4, we demonstrate the maximum BPL w.r.t. different ϵ and different transition matrices that represents P_i^B . In (a) and (b), the supremum does not exist. In (c) and (d), we can directly calculate the supremum using Theorem 5. The results are in line with the ones that computes BPL step by step at each time point using Algorithm 1.

Achieving α -DP $_{\mathcal{T}}$ by limiting upper bound. We now design a data release algorithm utilizing Theorem 5 to bound TPL. Theorem 5 tells us that, if it is not the strongest temporal correlation (i.e., $d = 0$ and $q = 1$), we may bound BPL or FPL within a desired value by allocating an appropriate privacy budget to a traditional DP mechanism at each time point. A problem is that, in Theorem 5, the q and d are assumed to be the ones that give the maximum value of the objective function; however, they are initially unknown. According to our analysis of Algorithm 1, such q and d depend on not only the given transition matrix but also the previous BPL (or the next FPL); however, the “previous BPL” is not clear when BPL achieves its supremum at some time point. To solve this problem, we can consider that, if T is close to infinite, BPL at time T and $T+1$ are both supremum, so that we can find q and d that give the maximum objective function using Algorithm 1 by setting the “previous BPL” to such supremum. Now, we can find an appropriate ϵ to bound BPL based on Theorem 5. For example, if $d = 0$ and $q \neq 1$, we can solve an equation with one variable ϵ : $\log \frac{(1-q)e^\epsilon}{1-qe^\epsilon} = \alpha^B$ (it is easy to prove that a positive solution always exists) where α^B is a desirable privacy level. Similarly, we can restrict FPL within a given value. We use this idea to bound both BPL and FPL, as shown in Algorithm 2.

Algorithm 2: Releasing Data with α -DP $_{\mathcal{T}}$ by upper bound

Input: P_i^B and P_i^F , $i \in U$; α (desired privacy level).
Output: private data satisfying α -DP $_{\mathcal{T}}$

```

1 foreach user  $i \in U$  do
2   Initialize  $\alpha^B \in (0, \alpha]$  as the supremum of BPL;
3   Find  $q^B$  and  $d^B$  that give the maximum value in Eq.(22) using
   Algorithm 1 with input  $\alpha_{t-1}^B = \alpha^B$ ;
4   Find  $\epsilon_i^B$  using Theorem 5 with the above  $q^B$ ,  $d^B$  and  $\alpha^B$ ;
5    $\alpha^F \leftarrow \alpha - \alpha^B + \epsilon_i^B$ ; //see Equation (10)
6   Find  $q^F$  and  $d^F$  using Algorithm 1 with input  $\alpha_{t+1}^F = \alpha^F$ 
7   Find  $\epsilon_i^F$  using Theorem 5 with the above  $q^F$ ,  $d^F$  and  $\alpha^F$ ;
8   if  $\epsilon_i^B < \epsilon_i^F$  then goto Line 2, initialize a larger  $\alpha^B$ ; else if
    $\epsilon_i^B > \epsilon_i^F$  then goto Line 2, initialize a smaller  $\alpha^B$ ; else
    $\epsilon_i \leftarrow \epsilon_i^B$ 
9  $\epsilon \leftarrow \min\{\epsilon_i, i \in U\}$ ;
10 return  $\epsilon$ -DP data at each time point
```

Achieving α -DP $_{\mathcal{T}}$ by leakage quantification. Algorithm 2 guarantees that the temporal privacy leakage never exceeds the given value, no matter how long the T is. However, when T is too short so that the accumulation of temporal privacy leakage have not resulted in a significant increase, we may not take the full advantage of the privacy budgets. Our observation is that, the DP mechanisms at the first and last time points should be allocated more budgets because they are relatively more “influential” in term of privacy loss. For example, BPL of $\mathcal{M}^t, t \in [2, T]$ is affected by the one of \mathcal{M}^1 , and FPL of $\mathcal{M}^t, t \in [1, T-1]$ is affected by \mathcal{M}^T . Our idea is to allocate more privacy budgets to \mathcal{M}^1 and \mathcal{M}^T so that both BPL and FPL are bounded in given values at each time point. For example, if we want that BPL at every time points are exactly the same value α^B , i.e., $BPL(\mathcal{M}^1) = \dots = BPL(\mathcal{M}^T) = \alpha^B$, then we need to make sure: (i) $PL_0(\mathcal{M}^1) = \alpha^B$ and (ii) $\mathcal{L}^B(\alpha^B) + \epsilon_m^B = \alpha^B$ in which ϵ_m^B is the privacy budget allocated in the “middle” of the timeline, i.e., from 2 to $T-1$. We can solve the above equations to obtain ϵ_m^B ensuring $BPL(\mathcal{M}^1) = \dots = BPL(\mathcal{M}^T) = \alpha^B$. Similarly, we can bound FPL in a given value by finding another ϵ_m^F . It is easy to know that, when $\epsilon_m^B = \epsilon_m^F$, we can exactly achieve α -DP $_{\mathcal{T}}$ at each time point.

We note that, the initializations of both α^B in Algorithm

Algorithm 3: Releasing Data with α -DP $_{\mathcal{T}}$ by quantification

Input: P_i^B and P_i^F , $i \in U$; α (desired privacy level).
Output: private data satisfying α -DP $_{\mathcal{T}}$

```

1 foreach user  $i \in U$  do
2   Initialize  $\alpha^B \in (0, \alpha]$  as the supremum of BPL,  $\epsilon_{i,1} = \alpha^B$ ;
3   Find  $q^B$  and  $d^B$  using Algorithm 1 with input  $\alpha_{t-1}^B = \epsilon_{i,1}$ 
4   Find  $\epsilon_{i,m}^B$  by solving  $\mathcal{L}^B(\epsilon_{i,1}) + \epsilon_{i,m}^B = \epsilon_{i,1}$  with  $q^B, d^B$ ;
5    $\epsilon_{i,T} \leftarrow \alpha - \epsilon_{i,1} + \epsilon_{i,m}^B$ ; //see Equation (10)
6   Find  $q^F$  and  $d^F$  using Algorithm 1 with input  $\alpha_{t+1}^F = \epsilon_{i,T}$ 
7   Find  $\epsilon_{i,m}^F$  by solving  $\mathcal{L}^F(\epsilon_{i,T}) + \epsilon_{i,m}^F = \epsilon_{i,T}$  with  $q^F, d^F$ ;
8   if  $\epsilon_{i,m}^B < \epsilon_{i,m}^F$  then goto Line 2, initialize a larger  $\alpha^B$ ; else if
    $\epsilon_{i,m}^B > \epsilon_{i,m}^F$  then goto Line 2, initialize a smaller  $\alpha^B$ ; else
    $\epsilon_{i,m} \leftarrow \epsilon_{i,m}^B$ 
9  $\epsilon_1 \leftarrow \min\{\epsilon_{i,1}, i \in U\}$ ,  $\epsilon_t \leftarrow \min\{\epsilon_{i,m}, i \in U\}$ ,  $\epsilon_T \leftarrow \min\{\epsilon_{i,T}, i \in U\}$ ;
10 return  $\epsilon_t$ -DP data at  $t \in [1, T]$ 
```

2 and 3 are nontrivial: too large or too small α^B results in more iterations to converge to $\epsilon_{i,m}^B = \epsilon_{i,m}^F$. We can prove that $\epsilon_{i,m}^B = \epsilon_{i,m}^F$ always can be achieved. We delegate the detailed descriptions and proofs to the long version of our paper.

VI. EXPERIMENTAL EVALUATION

In this section, we design experiments for the following: (1) verifying the runtime and correctness of our privacy leakage quantification algorithm (Algorithm 1), (2) investigating the impact of the temporal correlations on privacy leakage and (3) evaluating the data release Algorithms 2 and 3. We implemented all the algorithms in Java and conducted the experiments on a machine with an Intel Core i7 2.8GHz CPU and 16 GB RAM running OSX El Capitan.

The setting of temporal correlations. To evaluate if our privacy loss quantification algorithms can perform well under diverse circumstances, we need different degrees of temporal correlations. Although there are well studied methods to estimate the temporal correlations, in our experiments, we *generate* the correlations (transition matrices) directly to eliminate the effect of different estimation algorithms or datasets.

We now present a method for obtaining different degrees of temporal correlations. First, we generate a transition matrix indicating the “strongest” correlation that contains a cell with probability 1.0 at each row but for different columns (this type of transition matrix will lead to an upper bound of TPL, as shown in Examples 2 and 3). Then, we perform *Laplacian smoothing* [32], which is a method originally used to smooth a polygonal mesh, to *uniformize* the probabilities of P_i in different degree. Let p_{jk} be an element at the j th row and k th column of the matrix P_i . The new probabilities \hat{p}_{jk} are generated using Equation (24), where s is a positive parameter that controls the degrees of smoothing. A smaller s results in a stronger temporal correlation.

$$\hat{p}_{jk} = \frac{p_{jk} + s}{\sum_{u=1}^n (p_{ju} + s)} \quad (24)$$

We note that, the degrees of correlation with s are only comparable with each other under the same n (i.e., $|loc|$).

A. Runtime of Privacy Quantification Algorithms

In this section, we compare the runtime of our algorithm with Gurobi⁵ and lp_solve⁶, which are two well-known

⁵<http://www.gurobi.com/>. Commercial software. We use version 6.5.

⁶<http://lpsolve.sourceforge.net/>. Open source software. We use version 5.5.

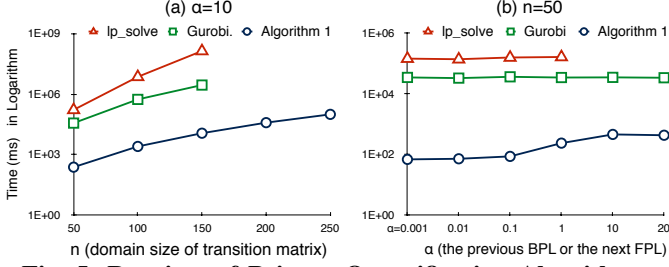


Fig. 5: Runtime of Privacy Quantification Algorithms.

softwares for solving optimization problems, e.g., the linear-fractional programming problem (17)~(19) in our setting. We run our algorithm 30 times and calculate its average runtime. For Gurobi and lp_solve , we run 5 times since they are very time-consuming. At each time, we randomly generate a transition matrix P_i whose elements are uniformly drawn from $[0, 1]$. We verified that the optimal solution returned by the three algorithms are the same. In the following, we compare the runtime. There are two factors that may affect the runtime: α as BPL at the previous time point or at the next time point (i.e., one input of Algorithm 1), n as the domain size of transition matrix. The results are shown in Figure 5.

Runtime vs. n . In Figure 5(a), we show the runtime of the three algorithms with inputs of $\alpha = 10$ and a $n \times n$ random probability matrix P_i . The runtime of all algorithms increase along with n because n is the number of variables in our linear-fractional program. Algorithm 1 significantly outperforms Gurobi and lp_solve . For example, in Figure 5(a), when $n = 150$, Algorithm 1 only spends 11 seconds, whereas the runtime of Gurobi and lp_solve are about 47 minutes and 38 hours, respectively. Since Gurobi and lp_solve spend tremendous time when $n > 150$, we omit to present them.

Runtime vs. α . In Figure 5(b), we show that, a larger previous BPL (or the next FPL), i.e., α , may lead to higher runtime of Algorithm 1, whereas Gurobi and lp_solve are stable for varying α . The reason is that, when α is large, Algorithm 1 may take more time in Lines 9 and 10 for updating each pair of $q_j \in \mathbf{q}^+$ and $d_j \in \mathbf{d}^+$ to satisfy Inequality (20). It is easy to see that more updates may occur due to a large α . However, such slight growth along with α will not last so long because it updates n times at most in Line 10. As shown in Figure 5(b), when $\alpha > 10$, the runtime of Algorithm 1 becomes stable. We only obtain a part of the runtime for lp_solve because a precision problem occurs when $\alpha \geq 10$ due to the design of lp_solve .

B. Impact of Temporal Correlations on Privacy Leakage

In this section, for the convenience of explanation, we only present the impact of temporal correlations on BPL because the growth of BPL and FPL are in the same way but in the reversed directions on the timeline. We examined s values in Equation (24) ranging from 0.005 to 1. We set n to 50 and 200. Let ϵ be the privacy budget of \mathcal{M}^t at each time point. We test $\epsilon = 1$ and 0.1. The results are shown in Figure 6 and are summarized as follows.

Privacy Leakage vs. s . Figure 6 shows that the privacy leakage caused by a non-trivial temporal correlation will increase over time, and such growth first increases sharply

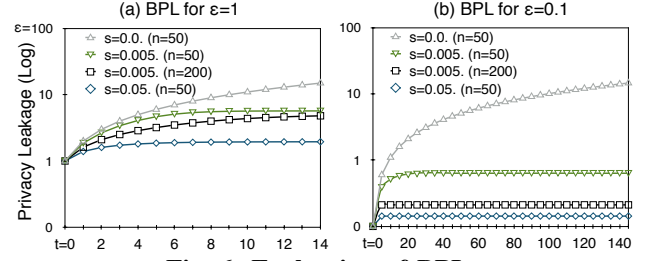


Fig. 6: Evaluation of BPL.

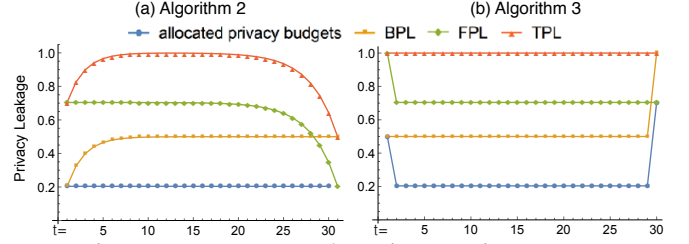


Fig. 7: Data Release Algorithms with 1- $DP_{\mathcal{T}}$.

and then remains stable because the increment is calculated recursively. The increase caused by a stronger temporal correlations (i.e., smaller s) is steeper, and the time for the increase is longer. Consequently, stronger correlations result in higher privacy leakage.

Privacy Leakage vs. ϵ . Comparing Figures 6(a) and (b), we found that 0.1-DP significantly delayed the growth of privacy leakage. Taking $s = 0.005$, for example, the noticeable increase continues for almost 8 timestamps when $\epsilon = 1$ (Figures 6(a)), whereas it continues for approximately 80 timestamps when $\epsilon = 0.1$ (Figures 6(b)). However, after a sufficient long time, the privacy leakage in the case of $\epsilon = 0.1$ is not substantially lower than that for the case of $\epsilon = 1$ under stronger temporal correlations because although the privacy leakage is eliminated at each time point by setting a small privacy budget, the adversaries can eventually learn sufficient information from the continuous releases.

Privacy Leakage vs. n . Under the same s , TPL is smaller when n (dimension of the transition matrix) is larger, as shown in the lines $s = 0.005$ with $n = 50$ and $n = 200$ of Figure 6. This is because the transition matrices tend to be uniform (weaker correlations) when the dimension is larger.

In conclusion, the experiments reveal that our quantification algorithms can flexibly respond to different degrees of temporal correlations.

C. Evaluation of Data Releasing Algorithms

In this section, we first show a visualization of privacy allocation of Algorithms 2 and 3, then we compare the data utility in terms of Laplace noise.

Figure 7 shows an example of budget allocation, w.r.t. $P_i^B = \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix}$ and $P_i^F = \begin{pmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{pmatrix}$. The goal is 1- $DP_{\mathcal{T}}$. It is easy to see that Algorithm 3 has better data utility because it exactly achieves the desired privacy level.

Figure 8 shows the data utility of Algorithms 2 and 3 with 2- $DP_{\mathcal{T}}$. We calculate the absolute value of the Laplace noise with the allocated budgets (as shown in Figure 7). Higher value

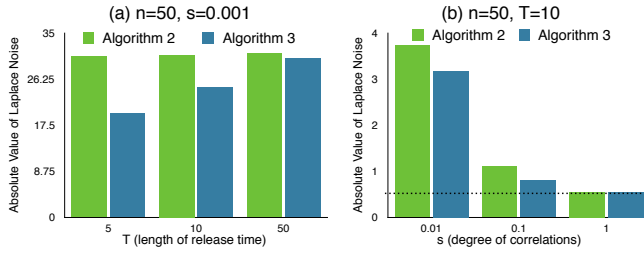


Fig. 8: Data utility of 2-DP_T mechanisms.

of noise indicate low data utility. In Figure 8(a), we test the data utility under backward and forward temporal correlation both with parameter $s = 0.001$, which means relatively strong correlation. It shows that, when T is short, Algorithm 3 outperforms Algorithm 2. In other words, whenever how long T is, Algorithm 2 perturb data in the same way. In Figure 8(b), we investigate the data utility under different degree of correlations. The dash line indicates the absolute value of Laplace noise if no temporal correlation exists (privacy budget is 2). It is easy to see that the data utility significantly decays under strong correlation $s = 0.01$.

VII. RELATED WORK

Several studies have questioned whether differential privacy is valid for correlated data. Kifer and Machanavajjhala [25] [26] [27] first raised the important issue that differential privacy may not guarantee privacy if adversaries know the data correlations. In their line of work, they [25] argued that it is not possible to ensure any utility in addition to privacy without making assumptions about the data-generating distribution and the background knowledge available to an adversary. To this end, they proposed a general and customizable privacy framework called *PufferFish*, in which the potential secrets, discriminative pairs, and data generation need to be explicitly defined. Yang et al. [36] further investigated differential privacy on correlated tuples described using a proposed Gaussian correlation model. The privacy leakage w.r.t. adversaries with specified prior knowledge can be efficiently computed. The major difference between correlation \mathbb{G} in this work and Gaussian correlation model \mathcal{G} in [36] is that the latter requires a connected graph (i.e., any user in the database should have at least one correlated user), whereas the \mathbb{G} is not necessarily connected (and is thus more practical).

Zhu et al. [38] proposed correlated differential privacy by redefining the sensitivity of queries on correlated data; however, the privacy guarantee provided by this definition for spatio-temporal data is unclear. Very recently, Liu et al. [9] proposed dependent differential privacy by introducing dependence coefficients for analyzing the sensitivity of different queries under probabilistic dependences between tuples. However, such dependence coefficients do not easily account for the spatio-temporal correlations.

Dwork et al. first studied *differential privacy under continual observation* and proposed event-level/user-level privacy [14] [16]. The previous studies in this setting focused on the problems of high dimension [1] [18] [34], infinite sequence [6] [8] [24], data sparsity [18], sliding window queries [5], and real-time publishing [19]. However, none of them addressed the problem of privacy loss under temporal correlations.

To the best of our knowledge, no study has reported the

risk of differential privacy under temporal correlations for the continuous aggregate release setting. Although a few studies [31] [35] have considered a similar adversarial model in which the adversaries have prior knowledge of temporal correlations represented by Markov chains, they focused on location privacy in the single-user setting. Shokri et al. [31] proposed an evaluation framework for location privacy protection, assuming that the adversary knows the transition probabilities of each user. Xiao et al. [35] proposed a mechanism extended DP for single user location sharing under temporal correlations modeled by Markov chains. In contrast, the scenario in this paper focuses on quantifying the privacy loss of traditional DP mechanisms under temporal correlations for continuous aggregate release setting.

VIII. CONCLUSIONS

In this paper, we quantified the risk of differential privacy under temporal correlations by formalizing, analyzing and calculating the privacy loss against adversary who has varying degrees of temporal correlations. This work opens up interesting future research directions, such as modeling temporal correlations with other type of correlations (e.g. tuple-wise correlations), and combining our methods with the previous studies that neglected the effect of temporal correlations in order to bound the temporal privacy leakage.

REFERENCES

- [1] G. Acs and C. Castelluccia. A case study: Privacy preserving release of spatio-temporal density in paris. In *KDD*, pages 1679–1688, 2014.
- [2] E. B. Bajalinov. *Linear-Fractional Programming Theory, Methods, Applications and Software*, volume 84. 2003.
- [3] J. Bolot, N. Fawaz, S. Muthukrishnan, A. Nikolov, and N. Taft. Private decayed predicate sums on streams. In *ICDT*, pages 284–295, 2013.
- [4] C. A. Bradley, H. Rolka, D. Walker, and J. Loonsk. BioSense: implementation of a national early event detection and situational awareness system. *MMWR supplements*, 54:11–19, 2005.
- [5] J. Cao, Q. Xiao, G. Ghinita, N. Li, E. Bertino, and K.-L. Tan. Efficient and accurate strategies for differentially-private sliding window queries. In *EDBT*, pages 191–202, 2013.
- [6] Y. Cao and M. Yoshikawa. Differentially private real-time data publishing over infinite trajectory streams. *IEICE Trans. Inf. & Syst.*, E99-D(1), 2016.
- [7] T.-H. H. Chan, M. Li, E. Shi, and W. Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *PETS*, pages 140–159, 2012.
- [8] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, 2011.
- [9] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. Dependence makes you vulnerable: Differential privacy under dependent tuples. In *NDSS*, 2016.
- [10] R. Chen, B. C. Fung, P. S. Yu, and B. C. Desai. Correlated network data publication via differential privacy. *VLDBJ*, 23(4):653–676, 2014.
- [11] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1998.
- [12] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.
- [13] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.
- [14] C. Dwork. Differential privacy in new settings. In *SODA*, pages 174–183, 2010.
- [15] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In S. Halevi and T. Rabin, editors, *Lecture Notes in Computer Science*, volume 3876, pages 265–284, 2006.
- [16] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *STOC*, pages 715–724, 2010.

- [17] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*, volume 9. 2013.
- [18] L. Fan, L. Xiong, and V. Sunderam. Differentially private multi-dimensional time series release for traffic monitoring. In *DBSec*, pages 33–48, 2013.
- [19] L. Fan, L. Xiong, and V. Sunderam. FAST: differentially private real-time aggregate monitor with filtering and adaptive sampling. In *SIGMOD*, pages 1065–1068, 2013.
- [20] Federal Highway Administration (FHWA). Traffic monitoring guide, 2013.
- [21] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez. Next place prediction using mobility markov chains. In *MPM*, pages 3:1–3:6, 2012.
- [22] T. Horanont, S. Phithakkitnukoon, T. W. Leong, Y. Sekimoto, and R. Shibasaki. Weather effects on the patterns of people’s everyday activities: A study using GPS traces of mobile phone users. *PLoS ONE*, 8(12):e81153, 2013.
- [23] Z. Jorgensen, T. Yu, and G. Cormode. Conservative or liberal? personalized differential privacy. In *ICDE*, pages 1023–1034, 2015.
- [24] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *PVLDB*, 7(12):1155–1166, 2014.
- [25] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *SIGMOD*, pages 193–204, 2011.
- [26] D. Kifer and A. Machanavajjhala. A rigorous and customizable framework for privacy. In *PODS*, pages 77–88, 2012.
- [27] D. Kifer and A. Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Trans. Database Syst.*, 39(1):3:1–3:36, 2014.
- [28] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD Explor. Newsl.*, 2(1):1–15, 2000.
- [29] W. Mathew, R. Raposo, and B. Martins. Predicting future locations with hidden markov models. In *UbiComp*, pages 911–918, 2012.
- [30] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD*, pages 19–30, 2009.
- [31] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux. Quantifying location privacy. In *SP*, pages 247–262, 2011.
- [32] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *SGP*, pages 175–184, 2004.
- [33] G. Theodorakopoulos, R. Shokri, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec. Prolonging the hide-and-seek game: Optimal trajectory privacy for location-based services. In *WPES*, pages 73–82, 2014.
- [34] Y. Xiao, J. Gardner, and L. Xiong. DPCube: releasing differentially private data cubes for health information. In *ICDE*, pages 1305–1308, 2012.
- [35] Y. Xiao and L. Xiong. Protecting locations with differential privacy under temporal correlations. In *CCS*, pages 1298–1309, 2015.
- [36] B. Yang, I. Sato, and H. Nakagawa. Bayesian differential privacy on correlated data. In *SIGMOD*, pages 747–762, 2015.
- [37] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *KDD*, pages 316–324, 2011.
- [38] T. Zhu, P. Xiong, G. Li, and W. Zhou. Correlated differential privacy: Hiding information in non-IID data set. *IEEE Trans. Inf. Forensics Security.*, 10(2):229–242, 2015.

APPENDIX A PROOF OF THEOREM 4

We need Dinkelbach’s Theorem and Lemma 3 in our proof.

Theorem 6 (Dinkelbach’s Theorem [12]). *In a linear-fractional programming problem, suppose that the variable vector is \mathbf{x} and the objective function is represented as $\frac{Q(\mathbf{x})}{D(\mathbf{x})}$. Vector \mathbf{x}^* is an optimal solution if and only if*

$$\max\{Q(\mathbf{x}) - \lambda * D(\mathbf{x})\} = 0 \text{ where } \lambda = \frac{Q(\mathbf{x}^*)}{D(\mathbf{x}^*)}. \quad (25)$$

Lemma 3. *For the following maximization problem ($k_1, \dots, k_n \in \mathbb{R}$) with the same constraints as the ones in the linear-fractional programming (17)~(19),*

$$\begin{aligned} & \text{maximize} && k_1 x_1 + \dots + k_n x_n \\ & \text{subject to} && e^{-\alpha_{t-1}^B} * x_k \leq x_j \leq e^{\alpha_{t-1}^B} * x_k, \\ & && 0 < x_j < 1 \text{ and } 0 < x_k < 1, \\ & && \text{where } x_j, x_k \in \mathbf{x}, j, k \in [1, n]. \end{aligned}$$

an optimal solution is as follows: if $k_i > 0$, let $x_i = e^{\alpha_{t-1}^B m}$ where m is a positive real number; if $k_i \leq 0$, let $x_i = m$.

Proof: Without loss of generality, we suppose that the smallest value in the optimal solution is x_n . Let y_j be $\frac{x_j}{x_n}$ for $j \in [1, n-1]$; then, $1 \leq y_j \leq e^{\alpha_{t-1}^B}$. Replacing x_j with y_j and setting $x_n = m$, we have a new objective function $\frac{1}{m} * (k_1 y_1 + \dots + k_{n-1} y_{n-1} + k_n)$ whose solution is equivalent to the original one. Because the only constraint is $1 \leq y_j \leq e^{\alpha_{t-1}^B}$, the following is an optimal solution for the maximum objective function: if $k_j > 0$, let $y_j = e^{\alpha_{t-1}^B}$; if $k_j \leq 0$, let $y_j = 1$. ■

Proof of Theorem 4: We first prove that, under the conditions shown in Theorem 4, i.e., Inequalities (20) and (21), an optimal solution of the problem (17)~(19) is:

$$\mathbf{x}^* = \begin{cases} x_j = e^{\alpha_{t-1}^B} * m & x_j \in \mathbf{x}^+ \\ x_k = m & x_k \in \mathbf{x}^- \end{cases}, \quad (26)$$

where m is a positive real number.

In combination with Dinkelbach’s Theorem, we rewrite our objective function as $\frac{Q(\mathbf{x})}{D(\mathbf{x})}$ in which $Q(\mathbf{x}) = \mathbf{q}\mathbf{x}$ and $D(\mathbf{x}) = \mathbf{d}\mathbf{x}$. Substituting \mathbf{x}^* of Equation (26) into $Q(\mathbf{x})$ and $D(\mathbf{x})$, we have $Q(\mathbf{x}^*) = q(e^{\alpha_{t-1}^B} - 1) + 1$ and $D(\mathbf{x}^*) = d(e^{\alpha_{t-1}^B} - 1) + 1$ (recall that $q = \sum q^+$ and $d = \sum d^+$). Then, we can rewrite Inequalities (20) and (21) in Theorem 4 as follows.

$$\frac{q_j}{d_j} > \frac{Q(\mathbf{x}^*)}{D(\mathbf{x}^*)}, \quad \forall j \in [1, n] \text{ where } q_j \in \mathbf{q}^+, d_j \in \mathbf{d}^+ \quad (27)$$

$$\frac{q_k}{d_k} \leq \frac{Q(\mathbf{x}^*)}{D(\mathbf{x}^*)}, \quad \forall k \in [1, n] \text{ where } q_k \in \mathbf{q}^-, d_k \in \mathbf{d}^- \quad (28)$$

Because $D(\mathbf{x}^*) > 0$, to prove \mathbf{x}^* in (26) is an optimal solution for $\frac{Q(\mathbf{x})}{D(\mathbf{x})}$, we only need to prove the maximum value of the following equation is equal to 0.

$$\text{maximize } \{D(\mathbf{x}^*)Q(\mathbf{x}) - Q(\mathbf{x}^*)D(\mathbf{x})\} = 0. \quad (29)$$

We expand the above equation as follows.

$$\begin{aligned} \text{Eqn. (29)} &= D(\mathbf{x}^*)(\mathbf{q}^+ \mathbf{x}^+ + \mathbf{q}^- \mathbf{x}^-) - Q(\mathbf{x}^*)(\mathbf{d}^+ \mathbf{x}^+ + \mathbf{d}^- \mathbf{x}^-) \\ &= (D(\mathbf{x}^*)\mathbf{q}^+ - Q(\mathbf{x}^*)\mathbf{d}^+) \mathbf{x}^+ + (D(\mathbf{x}^*)\mathbf{q}^- - Q(\mathbf{x}^*)\mathbf{d}^-) \mathbf{x}^- \end{aligned} \quad (30)$$

By Equations (27) and (28)), we have $D(\mathbf{x}^*)\mathbf{q}^+ - Q(\mathbf{x}^*)\mathbf{d}^+ > 0$ and $D(\mathbf{x}^*)\mathbf{q}^- - Q(\mathbf{x}^*)\mathbf{d}^- \leq 0$. Hence, according to Lemma 3, we can obtain the maximum value in Equation (29) by setting $\mathbf{x}^+ = [e^{\alpha_{t-1}^B} * m]$ and $\mathbf{x}^- = [m]$ where m is a positive real number. We obtain the maximum value in Equation (29).

$$\begin{aligned} & ((D(\mathbf{x}^*)q - Q(\mathbf{x}^*)d)e^\varepsilon m + (D(\mathbf{x}^*)(1-q) - Q(\mathbf{x}^*)(1-d)))m \\ &= (D(\mathbf{x}^*)(qe^\varepsilon + (1-q)) - Q(\mathbf{x}^*)(de^\varepsilon + (1-d)))m \\ &= (D(\mathbf{x}^*)Q(\mathbf{x}^*) - Q(\mathbf{x}^*)D(\mathbf{x}^*))m = 0 \end{aligned}$$

Therefore, by Dinkelbach’s Theorem, \mathbf{x}^* is an optimal solution for the problem (17)~(19). Substituting them into the objective function (17), we obtain the maximum value $\frac{q(e^{\alpha_{t-1}^B} - 1) + 1}{d(e^{\alpha_{t-1}^B} - 1) + 1}$. ■